МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Гусєв Євген Ігорович

УДК 004.04 (043.3)

Способи організації сумісного доступу до розподілених сторінок пам'яті в системах хмарних обчислень

Спеціальність 05.13.05 - Комп'ютерні системи та компоненти

Автореферат
дисертації на здобуття наукового ступеня
кандидата технічних наук

Київ - 2017

Дисертацією є рукопис.

Робота виконана на кафедрі обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Науковий керівник: доктор технічних наук, професор
Кулаков Юрій Олексійович
КПІ ім. Ігоря Сікорського, професор кафедри обчислювальної техніки.

Офіційні опоненти: доктор технічних наук, професор
Віноградов Микола Анатолійович
Національний авіаційний університет, Міністерство освіти і науки України, професор кафедри комп'ютерних інформаційних технологій;

кандидат технічних наук,
старший науковий співробітник
Чемерис Олександр Анатолійович,
Інститут проблем моделювання в енергетиці ім. Г. Є. Пухова НАН України,
в.о. зав. відділом.

Захист дисертації відбудеться «___» червня 2017 р. о 14:30 на засіданні спеціалізованої вченої ради Д 26.002.02 у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» (м. Київ, пр. Перемоги, 37, корп. 18, ауд. 516).

Відгуки на автореферат у двох екземплярах, завірені печаткою установи, просимо надсилати на адресу: 03056, м Київ, пр. Перемоги, 37, вченому секретарю Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

З дисертацією можна ознайомитись в бібліотеці Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

Автореферат розісланий «___» травня 2017 року

Вчений секретар
спеціалізованої ради,
кандидат технічних наук, доцент Орлова М.М.

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Сутність розв'язуваної в рамках дисертаційної роботи задачі полягає в оптимізації роботи зі спільними ресурсами в глобальних системах хмарних обчислень. В якості критеріїв оптимізації розглядаються метрики, які мають критичне значення в глобальних мережах, тобто таких мережах, в яких відносно невисокі показники часу відклику і пропускної спроможності. Це, в першу чергу - зниження ймовірності блокування ресурсу, зменшення часу блокування ресурсу і зниження міжвузлового трафіку. В роботі запропонований спосіб доступу до спільних ресурсів, що враховує специфіку обробки в середовищі хмарних обчислень, досліджені моделі трафіків в сучасних базах даних (БД), і проведено порівняння алгоритмів доступу і блокування з існуючими на прикладах досліджуваних моделей.

Актуальність роботи. За більш ніж 50 років історії систем керування базами даних (СКБД) було створено безліч алгоритмів доступу до спільного ресурсу та їх модифікацій. Але з появою хмарних обчислень проблема в СКБД вийшла на інший рівень. Використання існуючих способів блокування в сучасних хмарних системах призводять до значного збільшення часу обробки, і як наслідок, часу блокування. Збільшення часу блокування знижує стійкість системи до перевантажень, пов’язаних з доступом до спільних ресурсів. А це призводить до того, що масштабування методами, що використовувались у розподілених базах даних не забезпечує достатнього рівня ефективності. Результатом сформованої практики є ситуація, коли СКБД в системах хмарних обчислень стає спільним ресурсом, використовуваним усіма обчислювальними потужностями. При масштабуванні хмарної системи збільшення обчислювальних потужностей часто впирається в неможливість пропорційного збільшення потужності (масштабування) СКБД. Проблема вирішується нарощуванням апаратних ресурсів центральної СКБД або істотною зміною архітектури додатків для зниження інтенсивності звернень до бази даних. Все це обумовлює кризу в БД, у результаті якої з’явилися нові архітектури БД, тісно пов'язані додатками (NoSQL DB) для розвантаження бази даних від запитів, та спостерігається новий виток розвитку дорогих програмно-апаратних комплексів (Oracle Exadata, IMB DB2 PureScale on z/OS). Варто відзначити, що хмарні технології для обчислювальних задач (включаючи рендерінг, трансформацію тощо) привнесли спрощення адміністрування за рахунок консолідації та зниження вартості ресурсів. Однак для СКБД хмарні технології вирішують лише одну задачу – підвищення ефективності управління, в той час як вартість ресурсів, в них задіяних, лишається високою, а у деяких випадках навіть перевищує аналогічні за можливостями традиційні «нехмарні» сервіси. Таким чином, складається ситуація, коли вимоги до потужності при консолідації зростають, а можливості використовувати для консолідації дешеві хмарні технології відсутні. Власне, низькі можливості хмарних систем консолідувати системи зі спільними ресурсами і зумовили кризу в розвитку СКБД в контексті поширення використання систем хмарних обчислень.

Таким чином, тематика дисертаційної роботи, спрямована на розробку ефективних способів доступу до спільних ресурсів в глобальних хмарних системах актуальна, і становить науковий і практичний інтерес.

Зв'язок роботи з науковими програмами, планами, темами.

Робота виконувалася згідно з планами наукових досліджень на кафедрі

обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» в рамках науково-дослідних робіт:

- «Розробка теоретичних основ побудови високопродуктивних комп'ютерних систем з динамічним розпаралелюванням обчислювальних процесів» (державний реєстраційний номер 0111U002729 згідно з науковим напрямом «Розробка високопродуктивних багатокластерних обчислювальних систем»);
- «Методи організації моніторингових інформаційно-аналітичних систем науково-освітнього призначення на основ і високопродуктивних обчислювальних кластерних технологій» (державний реєстраційний номер 0109U000526).

Мета і задачі дослідження. Метою роботи є підвищення ефективності способів доступу до спільного ресурсу в системах хмарних обчислень. Для досягнення поставленої мети в роботі вирішуються такі задачі.

1. Аналіз існуючих способів доступу до спільних ресурсів, виявлення їх недоліків і знаходження шляхів можливого удосконалення в середовищі хмарних обчислень для формування математичної моделі системи розподіленої СКБД в контексті вирішення задачі доступу до спільного ресурсу.

2. Розробка математичної моделі процесу одночасного доступу до спільного ресурсу в розподілених СКБД.

3.Розробка способу організації одночасного доступу в розподілених СКБД.

4. Експериментальна перевірка запропонованої моделі, оцінка очікуваного ефекту від застосування запропонованого способу.

Об'єктом досліджень є процес організації сумісного доступу в розподіленій СКБД, яка функціонує в межах системи хмарних обчислень.

Предметом досліджень є способи і засоби блокування спільних ресурсів та оповіщення вузлів в системах розподілених хмарних обчислень.

Методи дослідження ґрунтуються на теорії ймовірностей, теорії множин, використанні методів моделювання та методів комбінаторного аналізу.

Наукова новизна одержаних результатів полягає у вирішенні завдань масштабування в хмарних СКБД, які працюють в середовищі з багатьма користувачами. Запропоновано та обґрунтовано способи, що відрізняються від відомих відходом від класичного блокування при визначенні необхідного набору даних, оптимізацією оповіщення вузлів і асинхронної реалізацією коригуючих дій при вирішенні конфліктів. Такий підхід дозволяє підвищити пропускну спроможність системи хмарних обчислень, знизити час відклику запиту, знизити ймовірність конфліктів, зумовлених блокуванням і підвищити стійкість системи до перевантаження, пов'язаного з блокуваннями.

Автором одержані наступні нові наукові результати.

1. Запропоновано і обґрунтовано математичну модель організації одночасного доступу до спільного ресурсу в системах хмарних обчислень, що дозволяє оцінити ефективність відомих методів доступу і виявити причини, що зумовлюють виникнення перевантаження.

2. Вперше був запропонований і обґрунтований метод двофазного виконання транзакції, який на відміну від відомих методів організації доступу до розподілених сторінок пам'яті, дозволяє виключити блокування спільного ресурсу в системах хмарних обчислень на етапі визначення необхідного набору даних. Це дозволяє істотно збільшити

пропускну спроможність систем хмарних обчислень і скоротити час відклику.

3. Запропоновано і обґрунтовано метод призначення обробника ресурсу в системах хмарних обчислень, що дозволяє при виникненні черги знизити кількість послідовних пересилань за рахунок збільшення паралельних. Обґрунтовано доцільність такого підходу і досліджена область застосування запропонованого способу.

4. Запропоновано і обґрунтовано комбінований підхід до вибору процедури доступу до спільного ресурсу, який, на відміну від відомих, дозволяє оптимізувати процедуру спільного доступу в залежності від інтенсивності транзакцій і кількості оброблюваних ресурсів в кожній з них.

5. Запропоновано спосіб конвеєризації фаз виконання транзакції, який, на відміну від послідовного виконання транзакції, дозволяє поєднати виконання окремих частин транзакції. А це дозволяє знизити ймовірність перевантаження.

Практичне значення одержаних результатів визначається тим, що

1. Запропонований спосіб доступу до спільного ресурсу дозволяє включати в хмарні системи ресурси, не пов'язані з основною хмарою швидким каналом комунікації.
2. Запропоновано модульну математичну модель розподіленої СКБД, що дозволяє моделювати обробку або блокування даних при дослідженні затримок, а також моделювати трафік в залежності від набору операцій і їх розподілу. В силу модульності вона може бути використана і для аналізу інших алгоритмів обробки даних, що реалізуються в хмарних СКБД, в тому числі при дослідженні на стійкість до перевантаження пов’язаного з блокуваннями в залежності від топології, параметрів мережі та трафіку.
3. Проведено аналіз синтетичного тесту продуктивності СКБД TPC-C, що широко застосовується у промисловості, показані слабкі сторони цього тесту при використанні в хмарних системах.

Особистий внесок здобувача. Основні результати отримані автором самостійно. У роботі [6], написаній у співавторстві автором запропонований алгоритм реалізації розподілених транзакцій, який виключає блокування спільного ресурсу при виконанні та фіксації транзакції.

Апробація результатів дисертації.

Основні результати роботи доповідалися і обговорювалися на:

- XVI міжнародній науковій конференції IAI-2016 ім. Т.А.Таран 18-20 травня 2016 року;
- XXIII Міжнародній конференції з автоматичного управління 22-23 вересня 2016 року м. Суми.

Публікації. За результатами виконаних досліджень опубліковано 8 наукових робіт, з яких 6 статей в фахових науково-технічних спеціалізованих виданнях, що включені до переліку наукових фахових видань (з них 5 статей, що входять до баз міжнародної наукової електронної бібліотеки eLIBRARY.RU) та 2 тези доповідей Міжнародних науково-технічних конференцій.

Структура та обсяг дисертації. Дисертаційна робота складається з вступу, чотирьох глав, висновків по кожній главі та загальних висновків по роботі в цілому, списку використаних літературних джерел (91 найменування). Повний обсяг дисертації – 156 сторінок, у тому числі 121 сторінок основного тексту, 11 рисунків, 19 таблиць.

ОСНОВНИЙ ЗМІСТ ДИСЕРТАЦІЇ

У першому розділі проведено аналіз існуючих способів доступу до спільних ресурсів в середовищі хмарних обчислень в залежності від базової архітектури. Розглянуто системи, що ґрунтуються на реплікації змін, системи, які використовують shared nothing, shared disk або shared everything архітектуру. В результаті порівняння ефективності масштабування систем обґрунтовано доцільність подальшого удосконалення способу доступу, який базується на shared everything архітектурі. В якості прототипної моделі для дослідження shared everything архітектури була обрана система Oracle Real Application Clusters (Oracle RAC), оскільки вона має низку істотних переваг перед іншими відомими системами і має широкий спектр впроваджень, що забезпечує поширеність системи і впливає на якість її підтримки. Серед існуючих проблем архітектури розглянуто перевантаження, тобто ситуацію, коли тривалість обробки заявки перевищує довжину черги, що з'являється протягом обробки. Обґрунтовано, що перевантаження є найбільш стримуючим фактором використання глобальних мереж в системах хмарних обчислень. Визначено, що дослідження перевантажень у системах, що використовують shared everything архітектуру, а також розробка способів, які дозволять їх уникнути, або значно знизити ймовірність появи, є найбільш актуальним питанням в контексті задачі масштабування СКБД у системах хмарних обчислень.

Другий розділ містить опис математичної моделі для дослідження хмарних СКБД, що використовують shared everything архітектуру. Запропонована модель складається з трьох основних компонентів, кожен з яких може мати різні реалізації: модель затримок, модель конфліктів і модель трафіку (ймовірнісна модель). При побудові моделі затримок розглядаються сценарії доступу до сторінки в залежності від її стану. Oracle RAC, взятий в якості прототипної моделі, передбачає 3 стану: XCUR – ексклюзивний поточний, використовується при внесенні змін до сторінки, (в глобальному кеші всіх вузлів системи тільки один вузол одночасно може мати XCUR стан) SCUR – розподілене поточне, використовується для отримання актуального стану при читанні, (необмежена кількість вузлів можуть одночасно мати сторінку в цьому стані) і CR – цілісне читання – стан на момент часу в минулому. Основна задача дослідження стосується взаємодії при конфлікті, і тому в якості досліджуваних трафіків обрані такі, в яких немає CR запитів. В таблиці 1 наведено затримки системи в залежності від ключових факторів;

Таблиця 1

Час доступу без очікувань черги в залежності від сценарію

Запит стану		Існує в глобальному кеші			
		Xcur		scur	
		л	г	л	г
майстор	xcur	0	$t_{net}+t_{send}$	0	$t_{net}+t_{send}$
	scur	0	$t_{net}+t_{send}$	0	$t_{net}+t_{send}$
немайст.	xcur	0	$2t_{net}+t_{send}$	$2t_{net}$	$2t_{net}+t_{send}$
	scur	0	$2t_{net}+t_{send}$	0	$2t_{net}+t_{send}$

де: t_{net} – час пересилання службового пакету; t_{send} – час пересилання сторінки пам'яті з даними. Крім того, передбачається, що розмір кешу досить великий, щоб внеском часу читання сторінки з диска у середній час доступу до сторінки можна було знехтувати. Це для багатьох промислових систем дійсно справедливо, оскільки системи оснащуються достатнім об'ємом оперативної пам'яті.

Наведемо формулу для середнього часу доступу до сторінки:

$$t_{acc.0} = \varphi\left(2t_{net} + t_{send} - \frac{t_{net}}{n}\right) + 2\frac{v_r v_w}{(v_r + v_w)^2}(1-\varphi)\left(1-\frac{1}{n}\right)t_{net} = \varphi\left(2t_{net} + t_{send} - \frac{t_{net}}{n}\right) + \beta t_{net} \qquad (1)$$

де: φ – коефіцієнт локального незакешування, а за умови кількості вузлів n, ймовірність вузла бути майстром – $1/n$. У формулі використовуються інтенсивності v_w та v_r, для оцінки ймовірності того, що після SCUR запиту, буде XCUR, причому та тому ж вузлі, на який до того прийшов SCUR, і в той же час цей вузол не є майстер-вузлом для цієї сторінки. Позначення β, яке, як видно з формули, залежить лише від кількості вузлів та співвідношення інтенсивностей v_w та v_r – це вплив майстер вузла на затримки доступу до сторінки. Використання інтенсивностей для оцінки ймовірностей допустиме, якщо поток запитів сторінок пуасонівський. Для такого потоку отримана формула для коефіцієнту локального незакешування φ:

$$\varphi = \frac{v_w(n-1)}{(nv_w + v_r)^2}(1-\varphi)\left(1-\frac{1}{n}\right)t_{net} = \varphi\left(2t_{net} + t_{send} - \frac{t_{net}}{n}\right) + \beta t_{net} \qquad (2)$$

Модель конфліктів, що розглядається далі, має 2 рівня конфліктів: конфлікт за сторінки і конфлікт за власне ресурси БД, що захищаються блокуваннями. В результаті аналізу затримок, зумовлених конфліктом за сторінки, в роботі отримана система рівнянь для знаходження $t_{p.queue}$ – часу очікування черги:

$$\begin{cases} t_{wpl} = t_{send} + t_{p.queue} \\ t_{p.queue} = \varphi(v_w + v_r)t_{wpl}t_{send} - \dfrac{t_{send}}{2}\left(1 - e^{-\varphi(v_w+v_r)t_{wpl}}\right) \end{cases} \qquad (3)$$

$t_{p.queue}$ – час очікування черги протягом конфлікту за доступ до сторінки; v_w та v_r – інтенсивності читання та внесення змін до сторінки; t_{wpl} – тривалість очікування сторінки; φ – коефіцієнт незакешування сторінки.

Моделюючи конфлікти відзначається, що конфлікт за ресурси виникає рідше ніж конфлікт за сторінки, однак утримується довше. Блокування не віддається до кінця транзакції на відміну від конфлікту за сторінки, де наступний запит сторінки може бути оброблений відразу після внесення змін або виявлення заблокованості ресурсу, в який ми намагаємось внести зміни. Оскільки конфлікт за ресурси утримується протягом транзакції, обчислюються затримки транзакції. А саме: для ресурсів, в які вносяться зміни – час звертання до об'єкту «черга» (у разі існування черги), повторний перезапит сторінки з ресурсом, що оброблюється (якщо під час очікування черги вузол втратив XCUR «блокування» на сторінку) та власне час очікуванні черги. Щодо читання сторінок, то у shared everything, що реалізована у Oracle RAC, ресурс не блокується, а використовується мультиверсійне читання, що вимагає враховувати затримки на реалізацію мультиверсійного читання, а це – додатковій запит сторінки з undo інформацією. Таким чином, затримки для кожного ресурсу транзакції:

$$t_i = t_{acc.i} + p_{rl.i} t_{rl} + \sigma_{w.i}\left(p_{inv.i} t_{acc.i} + t_{queue.i}\right) + \sigma_{r.i}\, p_{undo.i} t_{acc.i} \qquad (4)$$

Індекс *i*, в даному випадку показує, що ресурс відноситься до *i*-того класу. Під класом ресурсу розуміється множина ресурсів з однаковою інтенсивністю доступу до них; $t_{acc.i}$ – час доступу до сторінки ресурсу *i*-того класу; $p_{rl.i}$ – ймовірність того, що довелося звертатися до об'єкта «черга» на іншому вузлі; t_{rl} – час 2 пересилань: вузлу, що володіє «чергою» і відповідь від нього – підтвердження ексклюзивного блокування ресурсу, що запитується; $p_{undo.i}$ – це ймовірність того, що читаючи сторінку ресурсу *i*-того класу довелося запитувати з віддаленого вузла UNDO інформацію; $p_{inv.i}$ – це ймовірність повторного перезапиту сторінки з ресурсом *i*-того класу; $t_{queue.i}$ – затримки очікування черги до *i*-того класу ресурсів; $\sigma_{w.i}$ – визначає читання або запис здійснюється при обробці *i*-того класу ресурсу в транзакції (для запису $\sigma_{w.i}$=1, а для читання $\sigma_{w.i}$=0). Відповідні ймовірності обчислюються за формулами:

$$p_{rl.i} = \sum_{j \in TRRS_i} \left(1 - e^{-\left(1-\frac{1}{n}\right)\nu_{l.i} t_{wtl.j}}\right) \frac{\nu_{t.j}}{\sum_{j_2 \in TRRS_i} \nu_{t.j_2}} \qquad (5)$$

$$p_{undo.i} = \sum_{j \in TRRS_i} \left(1 - e^{-\left(1-\frac{1}{n}\right)\left(\frac{\nu_{r.i}}{\nu_{w.i}}\nu_{l.i}\right) t_{wtl.j}}\right) \frac{\nu_{t.j}}{\sum_{j_2 \in TRRS_i} \nu_{t.j_2}} \qquad (6)$$

$$p_{inv.i} = \sum_{j \in TRRS_i} \left(1 - e^{-\nu_{l.i} t_{wtl.j}} - \nu_{l.i} t_{wtl.j}\, e^{-\left(\nu_w + \nu_r\right) t_{queue.i} + \nu_{l.i} t_{wtl.j}}\right) \frac{\nu_{t.j}}{\sum_{j_2 \in TRRS_i} \nu_{t.j_2}} \qquad (7)$$

$$p_{queue.i} = \sum_{j \in TRRS_i} \left(\nu_{rl.i} t_{wrl}^2 - \left(1 - e^{-\nu_{rl.i} t_{wrl.j}}\right)\left(\frac{t_{wrl.j}}{2} + t_{net}\right)\right) \frac{\nu_{t.j}}{\sum_{j_2 \in TRRS_i} \nu_{t.j_2}} \qquad (8)$$

де: $TRRS_i$ – множина класів транзакцій, в які залучені ресурси *i*-того класу; $v_{t.j}$ – интенсивність *j*-того класу транзакції; $t_{wtl.j}$ – тривалість *j*-того класу транзакції. Транзакції належать одному класу, якщо в однаковому порядку оброблюють однаковий набір ресурсів однакових класів. Для зручності, ресурси індексуються літерою *i*, а транзакції – *j*. У підсумку, можемо об'єднати рівняння (4),(5),(6),(7) та (8) у систему, в якій : *TR* – множина усіх класів транзакцій трафіку; $TRRS_i$ – класи транзакцій, у яких використовується *i*-тий клас ресурсу; $RSTR_j$ – класи ресурсів, що використовуються у *j*-тому класі транзакції.

Для розв'язання системи (9) застосовується ітераційний підхід. На першій ітерації тривалості транзакцій знаходяться підставивши нуль у час очікування черги, та ймовірності $p_{rl.i}$, $p_{undo.i}$ та $p_{inv.i}$, що залежать від тривалості транзакцій. Отримавши набір тривалостей транзакцій зможемо порахувати довжини черг до відповідних ресурсів та відповідних ним ймовірностей, після чого знов підставимо отримані значення у формули

розрахунку тривалостей транзакцій.

$$
\begin{cases}
\forall j \in TR \rightarrow \\
t_{wtl.j} = \sum\limits_{\forall i \in RSTR_j} \left(t_{acc.i} + p_{rl.i} t_{rl} + \sigma_{w.i} \left(p_{inv.i} t_{acc.i} + t_{queue.i} \right) + \sigma_{r.i} p_{undo.i} t_{acc.0} \right) \\
\forall i \in RS \rightarrow \\
t_{queue.i} = \sum\limits_{\forall j \in TRRS_i} \left(\nu_{rl.i} t_{wtl.j}^2 - \left(1 - e^{-\nu_{rl.i} t_{wtl.j}}\right) \left(\frac{t_{wtl.j}}{2} + t_{net} \right) \right) \frac{\nu_{t.j}}{\sum\limits_{j_2 \in TRRS_i} \nu_{t.j_2}} \\
p_{inv.i} = \sum\limits_{\forall j \in TRRS_i} \left(1 - e^{-\nu_{l.i} t_{wtl.j}} - \nu_{l.i} t_{wtl.j} e^{-\left(\nu_{w.i} + \nu_{r.i}\right) t_{queue.i} + \nu_{l.i} t_{wtl.j}} \right) \frac{\nu_{t.j}}{\sum\limits_{j_2 \in TRRS_i} \nu_{t.j_2}} \\
p_{rl.i} = \sum\limits_{j \in TRRS_i} \left(1 - e^{-\left(1 - \frac{1}{n}\right) \nu_{l.i} t_{wtl.j}} \right) \frac{\nu_{t.j}}{\sum\limits_{j_2 \in TRRS_i} \nu_{t.j_2}} \\
p_{undo.i} = \sum\limits_{\forall j \in TRRS_i} \left(1 - e^{-\left(1 - \frac{1}{n}\right) \left(\frac{\nu_{r.i}}{\nu_{w.i}} \nu_{l.i} \right) t_{wtl.j}} \right) \frac{\nu_{t.j}}{\sum\limits_{j_2 \in TRRS_i} \nu_{t.j_2}}
\end{cases}
\qquad (9)
$$

На завершенні другого розділу даються загальні відомості про ймовірнісну модель, дослідження трафіків, яка докладно розглядається в 4 розділі. Також обґрунтовано допустимість моделювання потоків доступу до сторінок пуасонівським потоком у діапазоні максимальних та мінімальних значень.

Третій розділ присвячений опису нового запропонованого способу доступу до спільних ресурсів у системах хмарних обчислень, а також його дослідженню в рамках описаної раніше математичної моделі. Запропонований автором спосіб, що дозволяє вирішити проблему перевантаження, базується на 3 взаємопов'язаних методах. Перший метод – це метод двофазного виконання транзакції, другий – метод призначення обробника ресурсу і третій – конвеєризація фаз обробки.

Суть двофазного метода полягає в тому, щоб під час першої фази виконати транзакцію, використовуючи всі наявні на вузлі закешовані сторінки, для того щоб отримати список спільних ресурсів. Сторінки ці можуть знаходитись у неактуальному CR стані, і обробка першої фази не вимагає SCUR або XCUR блокування. В процесі виконання відбувається асинхронна відправка майстер вузлу запиту на відповідний ресурс, і актуальні версії сторінок фактично очікуються після виконання першої фази. За результатом першої фази ми маємо виконану транзакцію, проте можливо, що через неактуальність використаних ресурсів транзакція буде вимагати корегувальних дій. Перша і друга фази відділяються очікуванням актуальних версій сторінок, що

використовувались протягом першої фази. Під час другої фази звіряються актуальні сторінки з використаними, і у випадку, коли вони відрізняються, перевіряється, чи необхідні для виконання транзакції ще ресурси або достатньо сторінок, отриманих протягом першої фази. У випадку, якщо зміни в актуальних сторінках не вплинули на результат – друга фаза закінчена, і транзакція фіксується (commit). Якщо вплинули, але вся необхідна інформація є у отриманих сторінках – транзакція відкатується (rollback) та одразу не відпускаючи «блокувань» на ресурси виконується ще раз, але з актуальними даними. У разі недостатньої кількості ресурсів, неотримані ресурси запитуються ще раз (теж краще паралельно всі), після чого транзакція відкатується, виконується знов та фіксується.

Основною метою методу призначення обробника ресурсу є розвантаження черг, що виникають до ресурсу та його сторінок. Ідея методу така: якщо наростає черга – то необхідно знизити кількість пересилань між вузлами. Для цього вибирається вузол, який ексклюзивно буде вносити зміни в «гарячу» сторінку. Такий підхід дозволяє уникнути черг, які утворюються через збільшення часу відгуку внаслідок додавання до часу обробки часів пересилань між вузлами. Метод може застосовуватися як для розвантаження ресурсів, так і для розвантаження гарячих сторінок, але в даній роботі розглянута його реалізація для розвантаження гарячих сторінок. У метод також закладена можливість міграції з вузла, який вносить зміни, на інший вузол, для забезпечення динамічного балансування навантаження при конфліктах за сторінки різних ресурсів, що виникають одночасно. Запропонована автором послідовність обміну повідомленнями наступна:

1. При виникненні довгої черги майстер-вузол «гарячої» сторінки повинен перевести сторінку в режим розвантаження. Тобто режим, в якому для даної сторінки буде інша послідовність пересилань. Це висуває вимоги до майстер вузлу відстежувати історію повідомлень, що пересилаються, в іншому випадку черга почне розвантажуватися тільки після обслуговування запитів сторінок, які прийшли після моменту переведення в режим розвантаження.
2. На запити вузлів, які виконують обробку, XCUR або SCUR на «гарячу» сторінку, майстер-вузол дає відповідь, що сторінка у режимі розвантаження із зазначенням вузла, що призначений виконувати розвантаження (надалі вузел-хостер).
3. Вузол, що обробляє транзакцію, в межах якої виконує обробку сторінки, формує пакет дій, які треба зробити з його сторінкою (Наприклад, внести зміни у 3-й стовбець 15-го рядка, за умови що значення у 6-му стовбці цього рядка більше 100), та пересилає цей пакет вузлу-хостеру. Тут може бути багато варіантів щодо формату пакету дій та обмежень, але в даному випадку (оскільки OLTP) ми обмежуємось тим, що інструкції пакету дій мають ґрунтуватися на константах, що включені до пакету або даних, розташованих на тій же сторінці.
4. Хостер, отримавши пакет дій, ставить його в чергу дій над певною сторінкою. Паралельно з тим сторінки з черги обробляються.
5. Обробивши сторінку, вузол-хостер відправляє поточну версію сторінки вузлу, що створив для цієї версії пакет дій. Оскільки володіння XCUR залишається за вузлом-хостером – вузлу, що обробляє транзакцію, надсилається актуальна, але CR версія сторінки. На цьому етапі можлива оптимізація, яку ми при моделюванні

враховувати не будемо: замість пересилання всієї сторінки пересилаються зміни з моменту останнього кешування на вузлі, що обробляє транзакцію.

6. Оброблювальний вузол отримує сторінку в CR стані, як ніби він сам вніс в неї зміни і відразу ж відправив далі за запитами інших сторінок.

Пропонований автором метод конвеєризації фаз є розвитком застосовуваного методу двофазної обробки транзакцій. Суть методу полягає в тому, що в разі розбиття послідовності транзакцій на групи, які визначаються відсутністю в локальному кеші будь-яких версій відповідних сторінок на якомусь кроці транзакції, друга фаза для кожної попередньої групи може виконуватися паралельно з першою фазою наступної. Такий підхід дозволяє протягом очікування черги до ресурсу попередньої групи виконати першу фазу наступної групи та розпаралелити очікування ресурсів для двох (чи більше) груп зі зсувом, отримавши конвеєр. Підхід дозволяє виконати перезапит сторінок та провести коригувальні дії протягом найбільшого (з урахуванням тривалості зсуву конвеєра) очікування черги. В разі помилки, закладеної в оптимістичний підхід, на якому ґрунтується метод, вона також може бути оброблена протягом очікування цієї черги. Варто відзначити, що в разі, якщо послідовність транзакцій не розпадається на групи – то конвеєризації фаз немає.

При двофазному виконанні транзакції по іншому виглядають і затримки очікування сторінки і затримки очікування ресурсу. Фактично – це очікування максимального за тривалістю паралельного запиту. Наведемо формулу обчислення ймовірності k конфліктів при асинхронному виконанні за час t:

$$p_k(t) = \prod_{j=1}^{n}\left(\sum_{i=1}^{k} p_{ji}(t)\right) - \sum_{i=1}^{k-1} p_{ji}(t) \qquad (10)$$

де: p_{ik} –ймовірність k конфліктів за ресурси i-того класу; n – кількість ресурсів в транзакції. Відповідні $p_{ik}(t)$ обчислюються за формулою:

$$p_{ik}(t) = \frac{\left(\varphi_i \nu_{pl.i} t\right)^k}{k!} e^{-\nu_{pl.i} t} \qquad (11)$$

де: $\nu_{pl.i}$ – інтенсивність звернень до сторінок i-того ресурсу; а φ_i – коефіцієнт незакешування. Знаючи ймовірності, маємо систему рівнянь:

$$\begin{cases} t_{p.queue.i} = \sum_{k=1}^{\infty} k\, p_{ki}(t_{wpl.i}) - \frac{t_{send}}{2}\left(1 - p_0(t_{wpl.i})\right) \\ t_{wpl.i} = t_{send} + t_{p.queue.i} \end{cases} \qquad (12)$$

Для розв’язання системи (12) використовується ітераційний спосіб аналогічний тому, який використовувався при розв’язанні системи (3).

При оцінці затримок черги до ресурсів транзакції в роботі використовується схожий підхід, однак, на відміну від конфлікту за сторінки, де час очікування черги завжди кратний t_{send}, транзакції у трафіку мають різну тривалість. Спочатку треба визначити можливі тривалості транзакції, після чого знайти ймовірності цих тривалостей, і, нарешті, обчислити середню тривалість як суму добутків. Наведемо формулу для обчислення можливих тривалостей для транзакції j_2-ого класу:

$$d_{lj_2}=t_{wtl.j_1}\left(k-\frac{1}{2}\right)$$
$$j_1\in TRRS_i$$
$$i\in RSTR_{j_2} \qquad (13)$$

Індекс l вказує на порядковий номер тривалості, якщо їх впорядкувати за зростанням. З урахуванням тривалостей, формула середньої довжини черги у транзакціях j_3-ого класу:

$$t_{queue.j_3}=\sum_{l=1}^{\infty}\left(\Omega_{l\,j_3}-\Omega_{(l-1)\,j_3}\right)d_{lj_3} \qquad (14)$$

де кожне значення Ω_{lj3} знаходиться за формулою:

$$\Omega_{lj_3}=\prod_{\forall i\in RSTR_{j_3}}\left(\sum_{k=0}^{k\,t_{wtl.j_1}<d_{lj_3}}\sum_{j_1\in TRRS_i}\frac{\nu_{t.j_1}}{\sum_{j_2\in TRRS_i}\nu_{t.j_2}}\frac{\left(\nu_{l.i}t_{wtl.j_1}\right)^k}{k!}e^{-\nu_{l.j}t_{wtl.j_1}}\right)$$

Враховуючи, що значення p_{rl}, p_{undo}, p_{inv} та φ_i при двофазному виконанні транзакції теж розраховуються не для кожного ресурсу, а для всієї транзакції, отримуємо систему рівнянь:

$$\begin{cases}
\forall\, j\in TR\rightarrow \\
t_{wtl.j}=\left(\varphi_j+p_{inv.j}+p_{xcp.j}\right)t_{acc.0}+\left(1+p_{inv.j}+p_{xcp.j}\right)t_{p.queue.j}+p_{rl.j}t_{rl}+p_{undo.j}t_{acc.0}+t_{queue.j} \\
t_{queue.j}=\sum_{l=1}^{\infty}\left(\Omega_{lj}\left(RSTR_j\right)-\Omega_{l-1\,j}\left(RSTR_j\right)\right)d_{lj}\left(TRTR_j\right) \\
p_{inv.j}=1-\prod_{i\in RSTS_j}\left(\sigma_{w.i}\sum_{\forall\, j_2\in TRRS_i}\left(e^{-\nu_{l.i}t_{wtl.j_2}}+\nu_{l.i}t_{wtl.j_2}e^{-\left(\nu_{w.i}+\nu_{r.i}\right)t_{queue.j_2}+\nu_{l.i}t_{wtl.j_2}}\right)\frac{\nu_{t.j_2}}{\sum_{j_3\in TRRS_i}\nu_{t.j_3}}\right) \\
p_{rl.j}=1-\prod_{\forall i\in RSTS_j}\left(\sum_{j_2\in TRRS_i}\left(e^{-\left(1-\frac{1}{n}\right)\nu_{l.i}t_{wtl.j_2}}\right)\frac{\nu_{t.j_2}}{\sum_{j_3\in TRRS_i}\nu_{t.j_3}}\right) \\
p_{undo.j}=1-\prod_{i\in RSTS_j}\left(\sigma_{r.i}\sum_{\forall\, j_2\in TRRS_i}\left(e^{-\left(1-\frac{1}{n}\right)\left(\frac{\nu_{r.i}}{\nu_{w.i}}\nu_{l.i}\right)t_{wtl.j_2}}\frac{\nu_{t.j_2}}{\sum_{j_3\in TRRS_i}\nu_{t.j_3}}\right)\right)
\end{cases} \qquad (15)$$

де: TR – множина усіх класів транзакцій трафіку; $TRRS_i$ – класи транзакцій, у яких використовується i-тий клас ресурсу; $RSTR_i$ – класи ресурсів, що використовуються у j-тому класі транзакції. Обчислення системи рівнянь можливе ітераційно, так само як і для класичного shared everything.

При моделюванні конвеєризації фаз розглядається ситуація відсутності будь-якої версії необхідної сторінки при виконанні першої фази. При цьому система рівнянь набуває вигляду:

$$
\begin{cases}
S_{js} \in PARTS_{jz} \rightarrow \\
t_{wtl.j} = \sum\limits_{\forall S_s} \left((\varphi_s + p_{inv.s} + p_{xcp.s}) t_{acc.0} + (1 + p_{inv.s} + p_{xcp.s}) t_{p.queue.s} + \right. \\
\qquad \left. + p_{rl.s} t_{rl} + p_{undo.s} t_{acc.0} + t_{queue.s} \right) \\
t_{queue.s} = \sum\limits_{l=1}^{\infty} \left(\Omega_{ls}(S_{js}) - \Omega_{(l-1)s}(S_{js}) \right) d_{lj}(S_{js}) \\
p_{inv.j} = 1 - \prod\limits_{\substack{\forall i \in S_s \\ S_s \in PARTS_{jz}}} \sigma_{w.i} \left(\sum\limits_{\forall j_2 \in TRRS_i} \left(e^{-\nu_{l.i} t_{wtl.j_2}} + \nu_{l.i} t_{wtl.j_2} e^{-(\nu_{w.i} + \nu_{r.i}) t_{queue.j_2} + \nu_{l.i} t_{wtl.j_2}} \right) \frac{\nu_{t.j_2}}{\sum\limits_{j_3 \in TRRS_i} \nu_{t.j_3}} \right) \\
p_{rl.j} = 1 - \prod\limits_{\substack{\forall i \in S_s \\ S_s \in PARTS_{jz}}} \left(\sum\limits_{j_2 \in TRRS_i} \left(e^{-\left(1 - \frac{1}{n}\right) \nu_{l.i} t_{wtl.j_2}} \right) \frac{\nu_{t.j_2}}{\sum\limits_{j_3 \in TRRS_i} \nu_{t.j_3}} \right) \\
p_{undo.j} = 1 - \prod\limits_{\substack{\forall i \in S_s \\ S_s \in PARTS_{jz}}} \sigma_{r.i} \left(\sum\limits_{\forall j_2 \in TRRS_i} \left(e^{-\left(1 - \frac{1}{n}\right) \left(\frac{\nu_{r.i}}{\nu_{w.i}} \nu_{l.i}\right) t_{wtl.j_2}} \frac{\nu_{t.j_2}}{\sum\limits_{j_3 \in TRRS_i} \nu_{t.j_3}} \right) \right)
\end{cases} \qquad (16)
$$

де: множина $PARTS_{jz}$ – множина усіх частин, на які розпалась транзакція внаслідок відсутності сторінки якоїсь версії.

Система рівнянь при конвеєризації фаз:

$$
\begin{cases}
t_{wtl.j} = \left(\varphi_s t_{acc.0} + t_{p.queue.j} + p_{rl.j} t_{rl} + t_{queue.j} \right) + \\
\qquad + \Omega' \left(p_{inv.s} + p_{xcp.s} \right) \left(t_{acc.0} + t_{p.queue.s+1} \right) \\
t_{queue.j} = \sum\limits_{l=0}^{\infty} \left(\Omega_{lj}(RSTR_j) - \Omega_{l-1\,j}(RSTR_j) \right) d_{lj}(TRTR_j) \\
p_{inv.j} = 1 - \prod\limits_{\forall i \in RSTS_j} \sigma_{r.i} \left(\sum\limits_{\forall j \in TRRS_i} \left(e^{-\nu_{l.i} t_{wtl.j}} + \nu_{l.i} t_{wtl.j} e^{-(\nu_{w.i} + \nu_{r.i}) t_{queue.j} + \nu_{l.i} t_{wtl.j}} \right) \frac{\nu_{t.j_2}}{\sum\limits_{j_3 \in TRRS_i} \nu_{t.j_3}} \right) \\
p_{rl.j} = 1 - \prod\limits_{\forall i \in RSTS_j} \left(\sum\limits_{j_2 \in TRRS_i} \left(e^{-\left(1 - \frac{1}{n}\right) \nu_{l.i} t_{wtl.j_2}} \right) \frac{\nu_{t.j_2}}{\sum\limits_{j_3 \in TRRS_i} \nu_{t.j_3}} \right) \\
p_{undo.j} = 1 - \prod\limits_{\forall i \in RSTS_j} \sigma_{r.i} \left(\sum\limits_{\forall j_2 \in TRRS_i} \left(e^{-\left(1 - \frac{1}{n}\right) \left(\frac{\nu_{r.i}}{\nu_{w.i}} \nu_{l.i}\right) t_{wtl.j_2}} \frac{\nu_{t.j_2}}{\sum\limits_{j_3 \in TRRS_i} \nu_{t.j_3}} \right) \right)
\end{cases} \qquad (17)
$$

Система рівнянь (17) відрізняється від системи рівнянь (15) тим, що для розрахунку середньої тривалості черги можливі тривалості черги обраховуються за формулою:

$$\forall S_s \in PARTS_{jz} \exists \begin{cases} D_{js}\{d_{0\,js}, d_{1\,js}, d_{2\,js}, \ldots, d_{ljs}, \ldots\} \\ \forall (l_1 > l_2) \rightarrow (d_{l_1 s} > d_{l_2 s}) \end{cases} \rightarrow$$

$$\begin{cases} d_{ls} = t_{wtl.\,j}\left(k - \frac{1}{2}\right) + (s-1)(2t_{net} + 2t_{send}) \\ i \in S_s \\ j \in TRRS_i \end{cases} \qquad (18)$$

У рівнянні системи (17) для обчислення тривалості $t_{wtl.j}$ використовується ймовірність Ω', яка обчислюється за формулою:

$$\Omega' = \prod_{\forall i \in RSTR_i} \left(\sum_{\substack{k=0 \\ j_1 \in TRRS_i}}^{k\,t_{wtl.\,j_1+s} + (s(i)-1)(2t_{net}+2t_{send}) < 4(t_{net}+t_{send})} \frac{\nu_{t.\,j_1}}{\sum_{j_2 \in TRRS_i} \nu_{t.\,j_2}} \frac{\left(\nu_{l.\,i} t_{wtl.\,j_1}\right)^k}{k!} e^{-\nu_{l.\,j} t_{wtl.\,j_1}} \right) \qquad (19)$$

Порівняно з формулою (1) для класичного shared everything – формула тривалості обробки при використанні метода призначення обробника ресурсу не залежить від співвідношення інтенсивностей:

$$t_{acc} = 3t_{acc} + t_{send} - \frac{1}{n}(5t_{net} + t_{send}) \qquad (20)$$

Незважаючи на те, що час обробки збільшується порівняно з класичним підходом – методом знімається проблема перевантаження при обробці сторінок.

У четвертому розділі проведена експериментальна перевірка запропонованої моделі та оцінка очікуваного ефекту від застосування розроблених методів при дослідженні трафіків. Розглядається 3 види трафіків: так званий елементарний трафік, трафік проводок, і трафік TPC-C, який реалізує широко застосовуваний в IT індустрії тест TPC-C для визначення продуктивності OLTP систем. У якості головної вхідної метрики для всіх трафіків розглядається середня інтенсивність звернень. В залежності від розподілу класів транзакцій всередині трафіків та взаємозалежностей класів ресурсів, що складають класи транзакцій, визначаються інтенсивності перевантаження, тобто граничні інтенсивності для кожного трафіку. Протягом дослідження параметри моделі, що відповідають фізичним параметрам системи, фіксуються: часи t_{send} і t_{net}, будемо вважати рівними, емулюючи таким чином глобальну мережу як транспорт; кількість вузлів системи – 4; максимальна кількість рядків, що розташовані на одній сторінці – 100, але для трафіку проводок розглянемо і ситуацію малих сторінок (10 рядків на сторінці), показавши чутливість трафіків до розміру сторінки. Експериментально модель порівнюється з реально функціонуючим Oracle RAC. Результати наведені на рис. 1 та 2. На графіках відображено нормовані значення: по осі ординат – середню довжину транзакції, виміряну у пересиланнях, а по осі абсцис – нормовану інтенсивність, тобто виміряну в зверненнях за час одного пересилання – t_{send}, тобто в t_{send}^{-1}.

Незважаючи на нехтування часом обробки у порівнянні з часом пересилання, що вносить систематичну складову у похибку, відносна похибка різниці експериментальних даних і одержаних моделлю не перевищує 10% відносно реальних відповідних тривалостей у Oracle RAC.

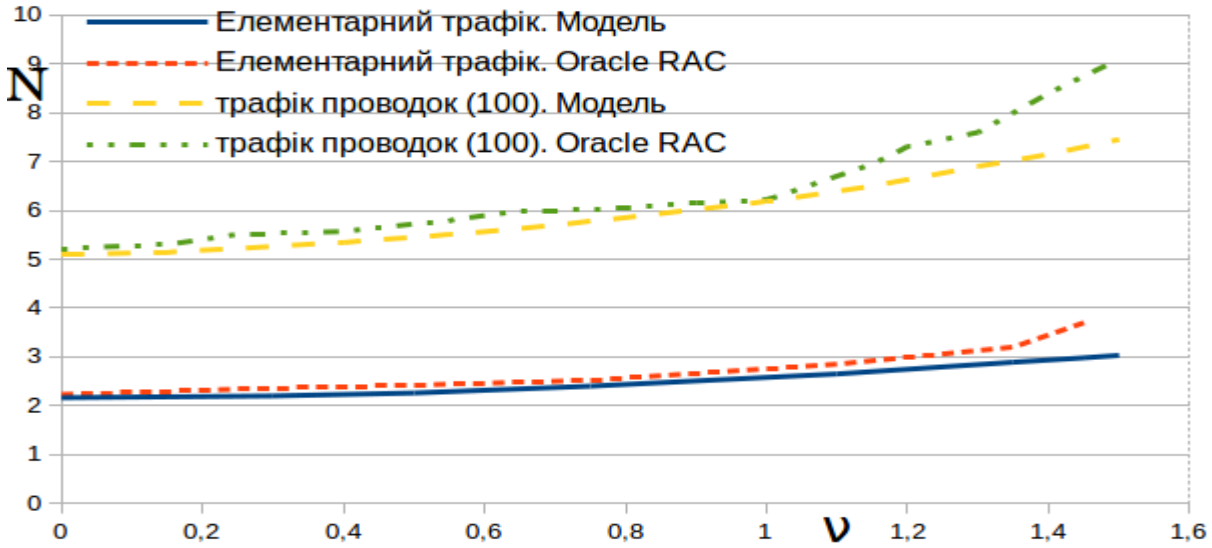

Рис. 1. Експериментальне порівняння моделі з прототипом

Також чітко простежується відповідність монотонності збільшення затримок обробки зі збільшенням інтенсивності запитів і відповідність характеру переходу системи в перевантаження. Більш ранній перехід в перевантаження для усіх трафіків в реальній системі зумовлений більш ранніми точками перевантаження, точність визначення моделлю для якого знаходиться в межах 10%, що означає прийнятність використання моделі для оцінки ефективності запропонованих методів.

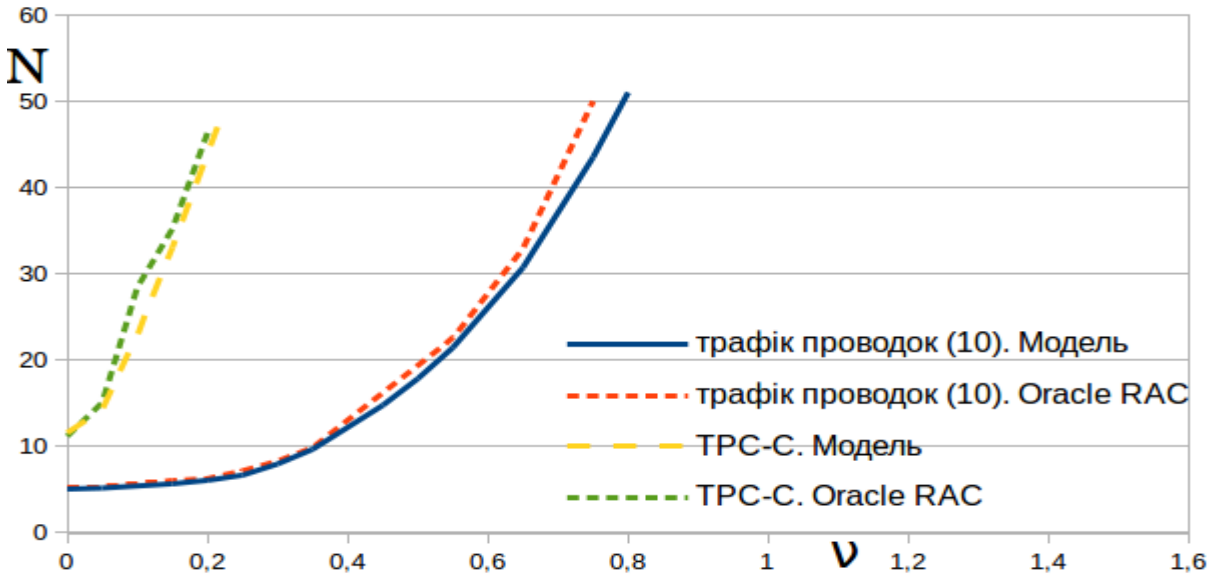

Рис. 2. Експериментальне порівняння моделі з прототипом

Порівнюючи класичний спосіб доступу в shared everything з таким, що використовує двофазне виконання транзакції, можна констатувати, що для двох прикладів метод виявився ефективним, а для двох – ні (рис. 3 та рис. 4). Це обумовлене тим, що перевантаження при зверненні до сторінок відбулося раніше, ніж перевантаження, пов’язане з блокуванням транзакції.

Метод призначення обробника ресурсу навпаки, для елементарного трафіку та трафіку проводок зі стандартним розміром сторінки виявився ефективним, в той час як для двох інших – ні. Також видно негативний ефект: гранична інтенсивність зменшилась при застосуванні методу для TPC-C трафіку та трафіку проводок зі стандартним розміром сторінки. Результати наведено на рис.5 та рис. 6.

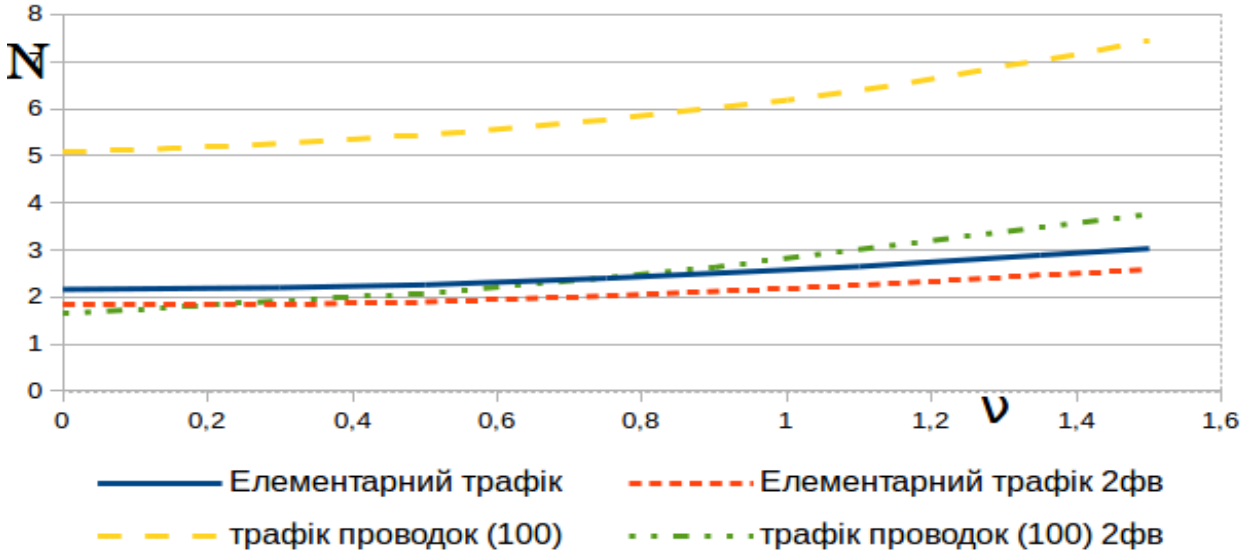

Рис.3. Метод двофазного виконання транзакції. Негативний ефект

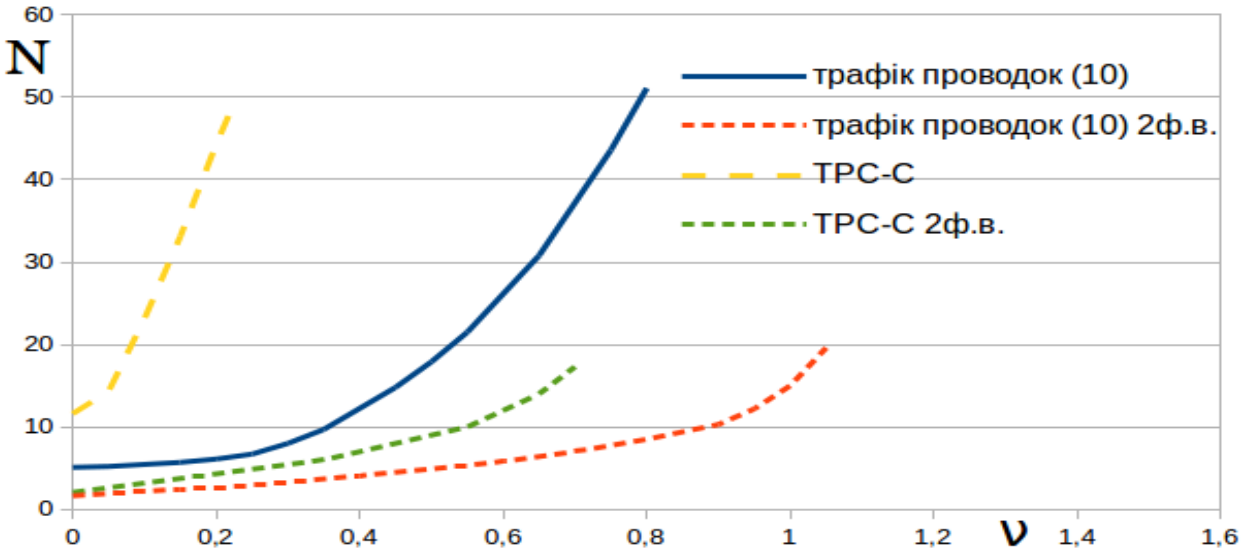

Рис.4. Метод двофазного виконання транзакції. Позитивний ефект

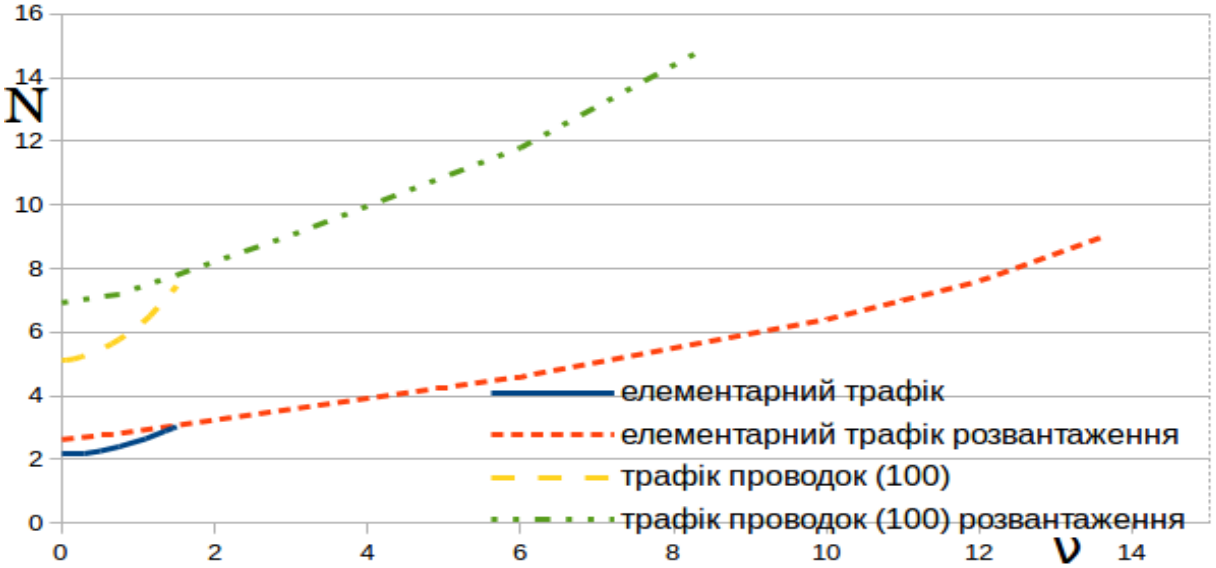

Рис.5. Метод призначення обробника ресурсу. Позитивний ефект

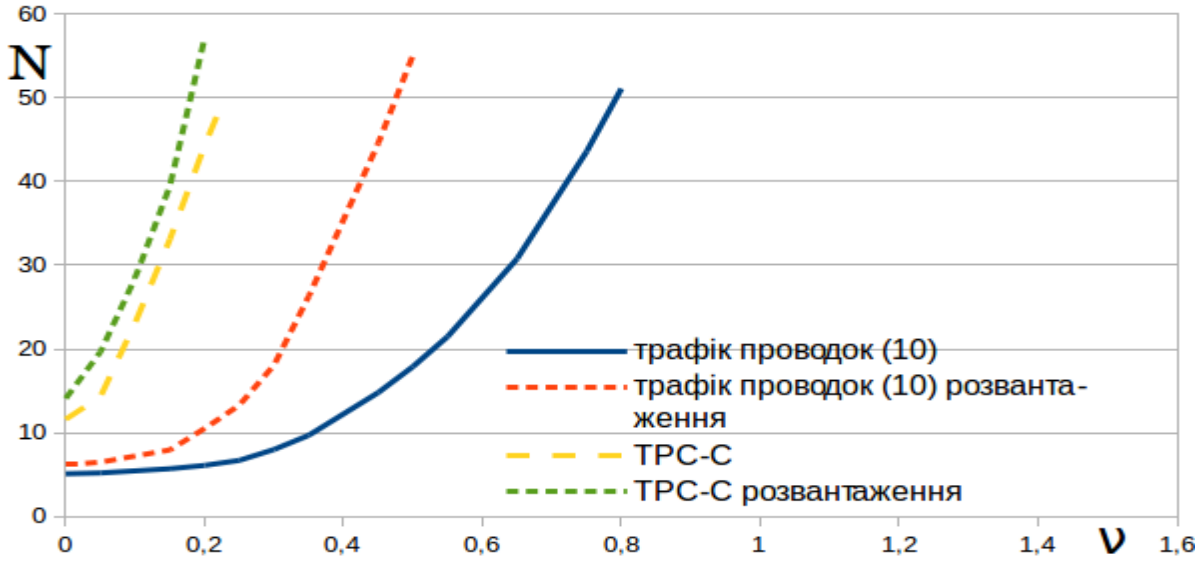

Рис.6. Метод призначення обробника ресурсу. Негативний ефект

Позитивний ефект від конвеєризації фаз можна отримати лише для багаторесурсних трафіків. Результати для TPC-C трафіку наведено на рис. 7:

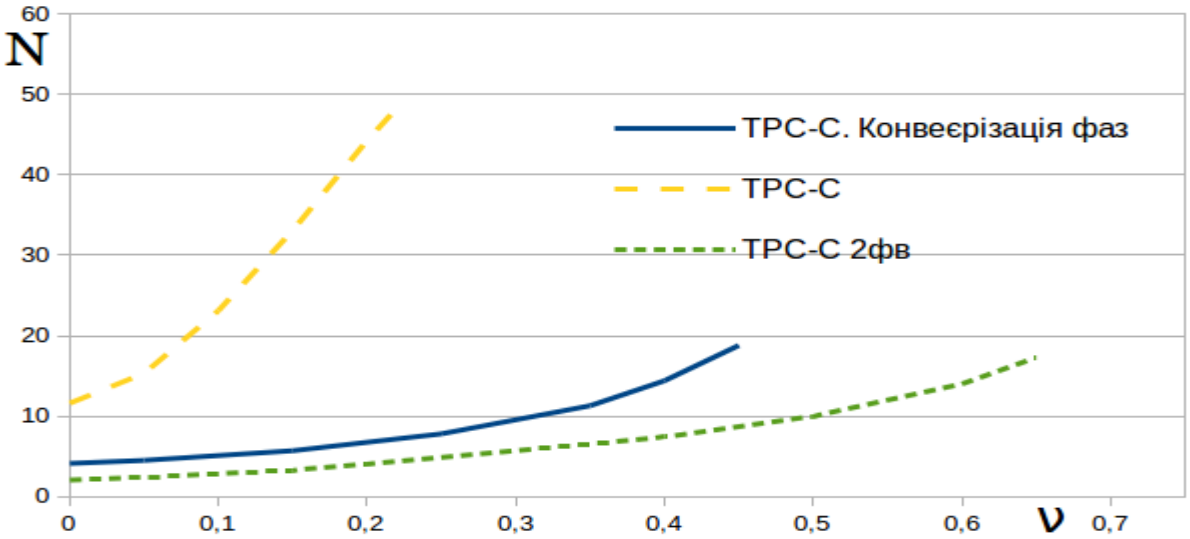

Рис.7. Конвеєризація фаз

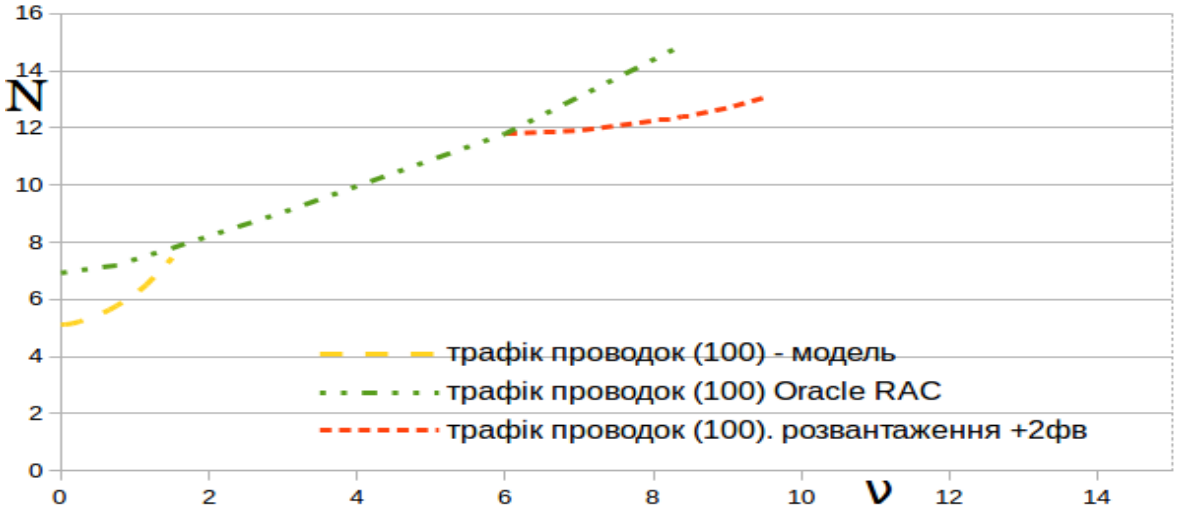

Рис.8. Комбінований підхід

Запропонований в роботі комбінований підхід включає в себе основні переваги всіх методів в залежності від інтервалу інтенсивності. Найбільш показовий приклад трафіку проводок зі стандартним розміром сторінки. Для нього у діапазоні (0;1.5) найбільш ефективний класичний спосіб доступу, у діапазоні (1.5;6,1) – призначення обробника ресурсу, а в (6;9.5) – комбінація призначення обробника ресурсу та двофазного виконання рис. 8.

У всіх випадках оцінка можливого ефекту була від 35% до 913%. Такий результат створює передумови для використання глобальних мереж в якості транспорту, але слід зазначити, що характеристики трафіку можуть значною мірою вплинути на ймовірність перевантаження. В той же час, суто інженерне рішення підвищити розмір сторінки для трафіка проводок підвищує ефективність способу з 35% до 650% (комбінований підхід). Наявність гарячого ресурсу у TPC-C трафіку (а це таблиця warehouse (склад), яка містить лише 2 рядки та оновлюється у 50% транзакцій, в яку вноситься час останнього оновлення складу) робить проблематичним масштабування в хмарному середовищі. Але навіть для такого випадку пропонований спосіб продемонстрував більш ніж 300% виграшу порівняно з традиційним. Таким чином, запропонований спосіб може посісти чинне місце в арсеналі засобів масштабування хмарних систем.

Основні наукові та практичні результати

У дисертаційній роботі виконано теоретичне обґрунтування і отримано нові рішення задачі організації доступу до спільного ресурсу. Для цього досліджені найбільш поширені системи, і, в якості прототипу моделі розподіленої хмарної СКБД, було обрано Oracle Real Application Clusters. Проаналізувавши недоліки прототипної моделі, були запропоновані нові методи організації спільного доступу, які знижують ймовірність перевантажень, пов’язаних з обробкою спільних ресурсів. Основні наукові і практичні результати полягають у наступному.

1. Виконано аналіз існуючих способів доступу до спільних ресурсів в середовищі хмарних обчислень, виявлені недоліки цих способів в контексті вирішуваних задач.

2. Запропоновано і обґрунтовано математичну модель організації одночасного доступу до спільного ресурсу в системах хмарних обчислень, що дозволяє оцінити ефективність відомих методів доступу і виявити причини, що зумовлюють виникнення перевантажень.

3. Вперше запропоновано і обґрунтовано метод двофазної обробки транзакції, який на відміну від відомих методів доступу, дозволяє виключити блокування спільного ресурсу в хмарних системах на етапі визначення необхідного набору даних. Це дозволяє істотно збільшити пропускну спроможність хмарних СКБД і скоротити час відклику.

4. Вперше запропоновано і обґрунтовано метод призначення обробника ресурсу, який за рахунок введення паралельних пересилань дозволяє виключити перевантаження, пов’язане з конфліктом за спільні сторінки.

5. Запропоновано метод конвеєризації фаз виконання транзакції, який дозволяє знизити ймовірність перевантаження за рахунок поєднання виконання окремих частин транзакції.

6. Запропоновано і обґрунтовано комбінований підхід до вибору процедури доступу до спільних ресурсів, який в залежності від інтенсивності транзакцій і кількості

оброблюваних ресурсів в транзакції дозволяє оптимізувати процедуру спільного доступу.

7. Проведено експериментальну перевірку запропонованої моделі та зроблена оцінка очікуваного ефекту від застосування розроблених методів.

ПУБЛІКАЦІЇ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Гусев Е.И. Моделирование способов организации доступа к распределённым страницам памяти в системах облачных вычислений основанных на shared everything архитектуре / Е.И. Гусев // Вісник НТУУ «КПІ» .Сер. Інформатика, управління та обчислювальна техніка. – 2016. – Вип. 64. – С.133-137.

Реферується науковометричною базою eLIBRARY.RU.

2. Гусев Е.И. Оптимизация доступа к распределённым страницам памяти в cloud computing системах основанных на shared everything архитектуре используя метод разгрузки очередей / Е.И. Гусев // Проблеми інформатизації та управління. – 2015. – Том 4, № 52. – С.17-21.

3. Гусев Е.И. Моделирование трафиков и оценка скорости распределённого доступа в системах облачных вычислений с общим ресурсом на примере Oracle RAC/ Е.И. Гусев // Вісник НТУУ «КПІ». Сер. Інформатика, управління та обчислювальна техніка. – 2015. – Вип. 62. – С. 32-42.

Реферується науковометричною базою eLIBRARY.RU.

4. Гусев Е.И. Математическое моделирование распределённой кластерной системы использующей shared everything подход (Oracle RAC) / Е.И. Гусев // Вісник НТУУ «КПІ». Сер. Інформатика, управління та обчислювальна техніка. – 2014. – Вип. 60. – С.106 – 113.

Реферується науковометричною базою eLIBRARY.RU.

5. Гусев Е.И. Исследование области применения неблокирующего алгоритма фиксации распределённых транзакций/ Е.И. Гусев, Кулаков А.Ю. // – Вісник НТУУ «КПІ». Сер. Інформатика, управління та обчислювальна техніка. – 2012. – Вип. 57. – С.76-80.

Реферується науковометричною базою eLIBRARY.RU.

6. Гусев Е.И. Исключение блокирования общего ресурса распределённых системах / Е.И. Гусев // – Вісник НТУУ «КПІ». Сер. Інформатика, управління та обчислювальна техніка. – 2011. – Вип. 54. – С.162 – 166. – *Здобувачем запропонований алгоритм реалізації розподілених транзакцій, який виключає блокування спільного ресурсу при виконанні та фіксації транзакції.*

Реферується науковометричною базою eLIBRARY.RU.

7. Гусев Е.И. Способы оптимизации совместного доступа к распределённым страницам памяти в системах облачных вичислений / Е.И. Гусев // матеріали XVI міжнародної наукової конференції IAI -2016 ім. Т.А. Таран 18-20 травня 2016 року. – 2016.

8. Гусев Е.И. Моделирование способов организации доступа к распределённым страницам памяти/ Е.И. Гусев // матеріали XXIII Міжнародної конференції з автоматичного управління «Автоматика 2016» 22-23 вересня 2016 року, м. Суми. – 2016.

АНОТАЦІЯ

Гусєв Є.І. Способи організації спільного доступу до розподілених сторінок пам'яті в системах хмарних обчислень. - Рукопис.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 - Комп'ютерні системи та компоненти. - Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Київ 2017.

В роботі запропоновано спосіб доступу до спільних ресурсів, який дозволяє виключити пов'язані з блокуванням перевантаження та знизити вимоги до каналів передачі даних. Його впровадження дозволить використовувати глобальну мережу як транспорт хмарних СКБД, внаслідок уникання перевантажень при доступі до спільних ресурсів. У роботі обґрунтовується необхідність скорочення часу відклику для зниження ймовірності перевантаження. Для вирішення проблеми використовуються три методи: метод двофазного виконання транзакції, метод конвеєризації фаз та метод призначення обробника ресурсу. Перші два використовують закешовані в процесі попередньої обробки версії сторінок для визначення списку необхідних спільних ресурсів на першій фазі, а під час другої фази повторно виконують транзакцію вже з актуальними сторінками. Метод конвеєризації фаз виконує фази в конвейері, що ефективніше для транзакцій зі слабозакешованими ресурсами. Метод призначення обробника ресурсу зменшує черги до сторінок, що інтенсивно оновлюються. Суть методу - фіксація для кожної «гарячої» сторінки вузлу обробки, для мінімізації кількості мережевих пересилань при обробці ресурсів в черзі.

Для оцінки ефекту способу розроблена математична модель, яка складається з моделі затримок, для оцінки тривалості доступу до сторінок пам'яті, моделі конфліктів – для оцінки ефекту блокувань, та модель трафіків для синтезу трафіків.

Ключові слова: хмарні обчислення, СКБД, Oracle RAC, перевантаження, блокування, черга, сторінка пам'яті, shared everything архітектура.

АННОТАЦИЯ

Гусев Е.И. Способы организации совместного доступа к распределённым страницам памяти в системах облачных вычислений. – Рукопись.

Диссертация на соискание ученой степени кандидата технических наук по специальности 05.13.05 - Компьютерные системы и компоненты. - Национального технического университета Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, 2017.

Диссертация посвящена проблеме оптимизации работы с общими ресурсами в

глобальных системах облачных вычислений. В условиях растущих требований к объёмам накопления, преобразования и обеспечения жизненного цикла информации, возникает тенденция по переносу баз данных в облачную среду. Особенности обработки общего ресурса в современных СУБД выдвигают повышенные требования по времени отклика к каналам передачи данных между узлами СУБД.

Предлагаемый в работе способ доступа к общим ресурсам позволяет исключить перегрузки, связанные с блокировками в облачной среде, снизить требования к каналам передачи данных при её развёртывании как по времени отклика, так и по пропускной способности. Применение предлагаемых новаций позволяет рассматривать в качестве транспорта в облачных СУБД не только высокопродуктивные коммутаторы, расположенные внутри одного ЦОДа, но и глобальную сеть, объединяющую ресурсы нескольких ЦОДов, в том числе распределённых географически.

Предлагаемый способ доступа является дополняющим к способу, используемому в современных облачных СУБД на базе shared everything архитектуры, что предполагает включение в себя традиционных алгоритмов доступа, успешно зарекомендовавших себя в реализации shared everything архитектуры (на примере Oracle RAC). Это позволяет динамически выбирать наиболее адекватный обрабатываемому трафику алгоритм как из набора традиционных алгоритмов, так и инновационных, предлагаемых в работе.

Основной акцент предлагаемых новаций посвящен проблеме возникновения перегрузок при доступе к общим ресурсам. В работе обосновывается необходимость сокращения времени отклика для снижения вероятности перегрузки. В качестве способов решения проблемы предлагаются 2 подхода, реализованные в трёх методах. Первый подход реализован в методе двухфазного выполнения транзакции и методе конвейеризации фаз. Предлагается использовать закешированные в процессе предыдущей обработки версии страниц при определении списка необходимых общих ресурсов. Использование этих страниц данных – оптимистический подход, поскольку страницы инвалидированы, и, возможно, именно из-за запрашиваемых данных. Но оптимистичность позволяет асинхронно разослать запросы на общие ресурсы, не выстраивая цепочку последовательных обращений, увеличивающих время отклика. Это целесообразно также и потому, что оценка времени определения взаимозависимости транзакций и ресурсов сопоставима с выполнением транзакции без её фиксации. Таким образом, первый раз (первая фаза) транзакция выполняется, используя неактуальные страницы. Завершающим этапом метода (второй фазой) является повторное выполнение транзакции с уже полученными актуальными страницами, а затем проверка и исправление ошибок, возникших из-за оптимистического подхода первой фазы для удовлетворения требований ACID. Различие методов, реализующих подход в том, что конвейеризация рассмотренных фаз показывает лучшую эффективность транзакций включающих слабо закешированные ресурсы. В качестве «платы» за выигрыш во времени выступает более чем двукратное увеличение объёма операций на каждом узле, что требует определения области целесообразности применения методов.

Второй подход – метод назначения обработчика ресурса – позволяет уменьшить возникающие при высоких интенсивностях обращения к страницам очереди к страницам. Эта проблема для традиционного способа доступа shared everything, реализованного в Oracle RAC, выдвигает высокие требования ко времени отклика между узлами, и, в контексте переноса в облачную среду, является ключевой. Суть метода –

зафиксировать для каждой «горячей» страницы узел обработки, минимизировав таким образом количество сетевых пересылок при последовательной обработке ресурсов в очереди. Негативным эффектом метода является увеличение объёма межузлового трафика.

Для оценки эффекта способа в контексте исследования области целесообразности автором предложена трёхкомпонентная математическая модель. Первый компонент – модель задержек, описывающая сценарии обработки страницы памяти необходима для вычисления времен обработки страницы. Второй компонент – модель конфликтов отражает существующий в shared everything архитектуре (на примере Oracle RAC) способ блокирования ОР в контексте взаимосвязей этих ресурсов с транзакциями. Модель конфликтов предполагает стационарность потока входящих заявок, разбиваемых на последовательности обработки. И третьим компонентом является модель трафика, главной целью которой является определение входящих параметров для модели конфликтов и модели задержек. В работе рассматриваются 3 вида трафиков – первые 2 предложены автором для наиболее полного освещения особенностей предлагаемых новаций, а 3й является синтетическим тестом (TPC BENCHMARK Standard Specification Revision 5.11) широко используемым для определения производительности программно-аппаратных комплексов в ИТ индустрии.

Ключевые слова: облачные вычисления, СУБД, Oracle RAC, перегрузка, блокировка, очередь, страница памяти, shared everything архитектура.

ABSTRACT

Gusev E.I. Techniques of shared access organization to distributed memory pages in cloud computing systems. - Manuscript.

Thesis for a Candidate of Technical Sciences degree in specialty 05.13.05 - Computer systems and components. - National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kiev, 2017.

The thesis is devoted to the problem of shared resource processing optimization in a cloud computing systems. There is proposed access technique to shared resource, which allows exclude the overloadings, associated with locks in the cloud environment and reduce the requirements for data transfer channels. The paper substantiates the need to reduce response time for reducing overloading probability. To solve the problem are used three methods: the method of two-phase transaction execution, method of piping phases and method of resource handler assigning. The first two use cached old versions of pages to determine required list of resources during first phase, and reexecute transaction on second phase to confirm ACID. The piping phase method is more efficient for transactions with low-cached resources due to piping approach. Method of resource handler assigning reduces queues to intensively updated resource pages. The key idea of method is fixing for each "hot" page processing node, thereby minimizing the number of network hops during page access.

To estimate the effect of the method the author offers three-component mathematical model consisting of: delay models to determine the duration of access to memory page, the model of conflict - to evaluate the effect of blocking, and traffic model – to synthesize traffic.

Keywords: cloud computing, DBMS, Oracle RAC, overloading, locks, queue, page of memory, shared everything architecture.